(2)

REPORT DOCUMENTATION P

AD-A257 040

1. AGENCY USE ONLY (Leave Blank)

2. REPORT DATE: August 1992

4. TITLE AND SUBTITLE: Automated Interpretation of Topographic Maps

5. FUNDING NUMBERS

6. AUTHOR(S): Cronin, T. M.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES):
U.S. Army CECOM Signals Warfare Directorate
Vint Hill Farms Station
Warrenton, VA 22186-5100

8. PERFORMING ORGANIZATION REPORT NUMBER: 92-TRF-0041

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT: Statement A: Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
Some new results which impact favorably upon the issue of automated topographic map interpretation are presented. Actually, the "new" results consist of heretofore undiscovered applications of two concepts known to mathematicians and computer scientists for many years: binary search, and the normal vector. Binary search is extended by the technique from not only one to two dimensions, but arguably to three dimensions, since topographical maps are two-dimensional representations of three-dimensional surfaces. Such maps are essentially sorted hierarchies of nested contours, which form a muliply-connected subdivision of the plane. The perimeter of a subdivision element is defined by a set of contours of extremal elevation and the edges of the map; a naming convention attaches a label to each element of the planar subdivision. Whenever one is afforded the luxury of dealing with a sorted data structure, one may invoke the power of binary search to achieve O[log n] time complexity during processing of a topographical query, where n is the number of contours which comprise a specific element of the planar subdivision. A topographical query is a request by a user to interpret the position of an arbitrary map coordinate, called query point, topographic map background.

14. SUBJECT TERMS: Artificial Intelligence, Applied Mathematics, Data Fusion, Topographic Line Map (TLM)

15. NUMBER OF PAGES: 21

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540 01-280-5500

Standard Form 298

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to **stay within the lines to meet optical scanning requirements.**

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - DOD - Leave blank.
DOE - DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA - NASA - Leave blank.
NTIS - NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Automated Interpretation of Topographic Maps

T. Cronin
CECOM Signals Warfare Directorate
Warrenton VA 22186-5100

Abstract: Some new results which impact favorably upon the issue of automated topographic map interpretation are presented. Actually, the "new" results consist of heretofore undiscovered applications of two concepts known to mathematicians and computer scientists for many years: binary search, and the normal vector. Binary search is extended by the technique from not only one to two dimensions, but arguably to three dimensions, since topographical maps are two-dimensional representations of three-dimensional surfaces. Such maps are essentially sorted hierarchies of nested contours, which form a multiply-connected subdivision of the plane. The perimeter of a subdivision element is defined by a set of contours of extremal elevation and the edges of the map; a naming convention attaches a label to each element of the planar subdivision. Whenever one is afforded the luxury of dealing with a sorted data structure, one may invoke the power of binary search to achieve O[log n] time complexity during processing of a topographical query, where n is the number of contours which comprise a specific element of the planar subdivision. A topographical query is a request by a user to interpret the position of an arbitrary map coordinate, called the query point, in the context of a topographic map background. An "interpretation" as currently defined consists of a five-tuple of information: the label of the map subdivision element within which the query point resides; the topographical contour of the subdivision element within which the point minimally resides; the local slope at the point; the local elevation at the point; and the flank of the partition element (hillside) upon which the point is situated. As an example, the following list is an interpretation of a query point: (Mt. Hood subdivision, contour #10600-d, 65 degree gradient, 10655 feet elevation, 350 deg NW). As is the case with one-dimensional binary search, the two-dimensional version must concern itself with items having identical keys. Because topographical maps may contain multiple contours lying at the same elevation, the topographical query process must have a mechanism for choosing among them before proceeding. Thus, when "halving the interval", one must check to ensure that the interval is in fact uniquely defined. Inclusion testing within contours is achieved with a deterministic point-in-polygon algorithm developed by the Army in previous research. The traditional normal vector is utilized extensively by the point-in-polygon algorithm, and also by the processing components which interpolate slope and elevation, and determine hillside emplacement. A new theorem derived from the law of cosines provides a decision rule based on integer arithmetic to decide which segments of a polygonal boundary justify a computation of the normal vector. As a byproduct of the research, an algorithm based on the Cevian formula has been developed to find the nearest segment to a query point, without using any floating point operations whatsoever. Future research issues include the topic of visual line-of-sight (binary map coloring) from a query point, and an analysis of the space and time complexity of the new technique, vs. the more burdensome alternative of storing elevation and slope data in large raster archives.

I. BACKGROUND AND TERMINOLOGY

Topographic Line Maps.

A topographic line map (TLM), also known as a contour map, is a vector representation of the boundaries of cross sections of the earth's surface. The projection of the boundary of a cross section of the earth's surface onto the xy-plane is called a topographic *contour*. The equispacing between cross sections along the z axis is called the *contour interval* of the map. A contour map is bounded by the rectangular edges of a map sheet. The edges of a map form what is called a *clipping region*, which prevents an observer from knowing the behavior of portions of contours which pass outside the rectangular perimeter of the map.

Heuristics to guide map understanding soon become apparent to a novice: e.g., when contours are closely packed together in the xy-plane, the underlying terrain is steep; when far apart, the terrain is flat, or of gentle gradient. As a general rule, contours do not cross, although there are rare exceptions such as natural bridges, overhangs, or bizarre sandstone formations like those found in Utah. For those well-versed in their interpretation, topographic maps can provide a realistic portrayal of actual terrain. However, many human beings find topographic maps difficult to interpret, and have problems visualizing terrain from contour data. It is for this reason that the Army Topographic Engineering Center has opted to utilize perspective displays as an alternative to topographic maps [T1]. Perspective displays render an artistic version of a terrain as it appears from some vantage point near or upon the ground. Figure 1 is a graphic borrowed from reference [K4], and portrays a perspective display of a terrain, together with its corresponding topographic contour representation.

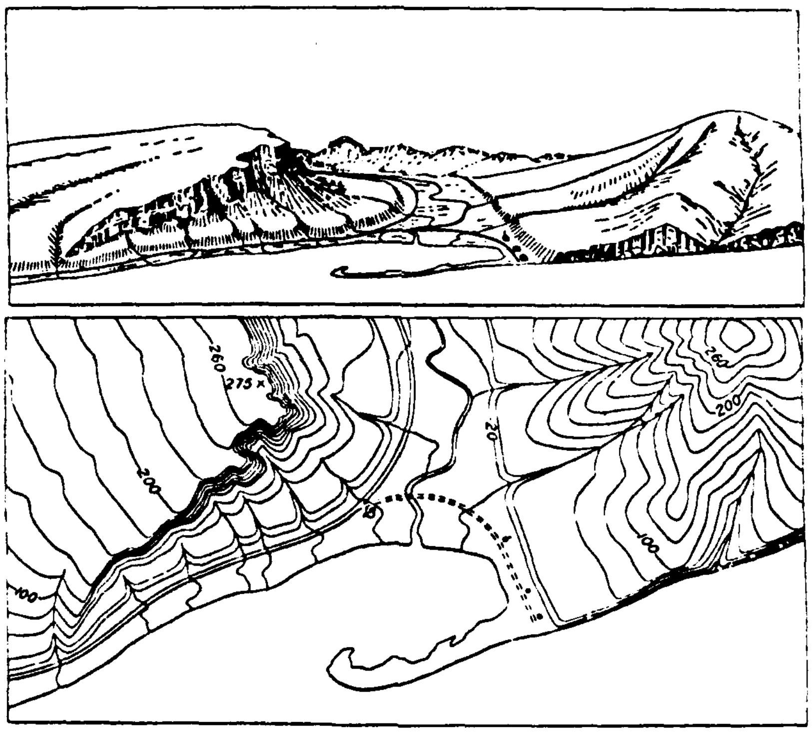

Figure 1. A perspective display and its corresponding topographic line map.

Interpreting a Topographic Line Map.

The objective of automated topographic map interpretation is to write a computer algorithm which locally describes a random query point in the context of a contour map. An interpretation as currently defined consists of five pieces of information: the label (if it exists) of the subdivision within which the query point resides; the name of the topographic contour which brackets the query point from outside; the elevation above or below sea level at the query point; the slope of the terrain at the query point; and the directional gradient at the query point. As an example, the following list constitutes an interpretation of a query point: (Mt. Hood subdivision, contour #10600-d, 65 degree gradient, 10655 feet elevation, 350 deg NW).

The five pieces of information currently sought by the algorithm hardly constitute a complete interpretation of a query point. Other descriptors are desirable in the long term: e.g., the visual line-of-sight from the query point, the profile of a traversible path passing through the query point, the profile of a path which optimally avoids the query point, the feasibility of using the query point as a site for sensor placement, etc. However, the five primitive data currently being returned by the algorithm go far toward providing inputs to some of the higher level queries, which may be synthetically constructed from the primitive queries.

Contour Notation.

In this section, notation is adopted to facilitate reasoning with topographic contours. Associated with every contour is a specific value, denoted El(C), which represents the contour's elevation above or below sea level. The elevation is modulo k, where k is the fixed contour interval of the tcpographic map. On a particular map, there may be several distinct contours with the same elevation value; for spatial reasoning applications it is important to differentiate among them. Each contour with an elevation value above sea level is contained within another contour, and may itself contain contours.

If contour C_1 is contained within contour C_2, then C_1 is said to be *nested* within C_2. A contour cannot be contained within two or more contours which are not nested, but it *can* contain multiple contours which are not nested. If C is a contour of interest, then we denote the contour which minimally contains C to be C^-. By minimally contained, it is meant that any other contour D other than C^- which contains C also contains C^-, which implies that both C^- and C are nested within D. A contour minimally contained by C is called C^+, where the set of all such contours is denoted $\{C^+\}$. If a query point lies between two elements of $\{C^+\}$, then it is said to be a *saddle point*. Note that $\{C^+\}$ may be the null set. A graphic illustrating these concepts is shown below.

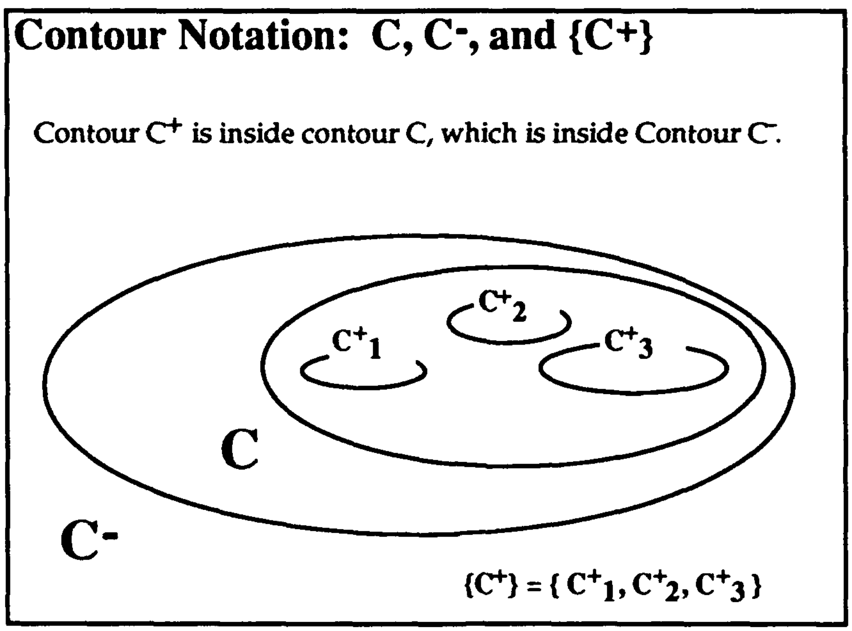

Figure 2. An illustration of contour containment.

A query point and its bracketing contours.

A query point is defined to be any random point of interest. Except for special cases, a query point is bracketed by adjacent contours of a map: one contour which encloses it, called the *outer bracket*, and another contour which does not enclose it, called the *inner bracket*. Since contours are well-ordered at equispaced elevations, the difference in elevation between bracketing contours is equal to plus or minus the fixed contour interval of the map (except for a zero difference at saddle or culvert points). The figure below illustrates the brackets of a query point.

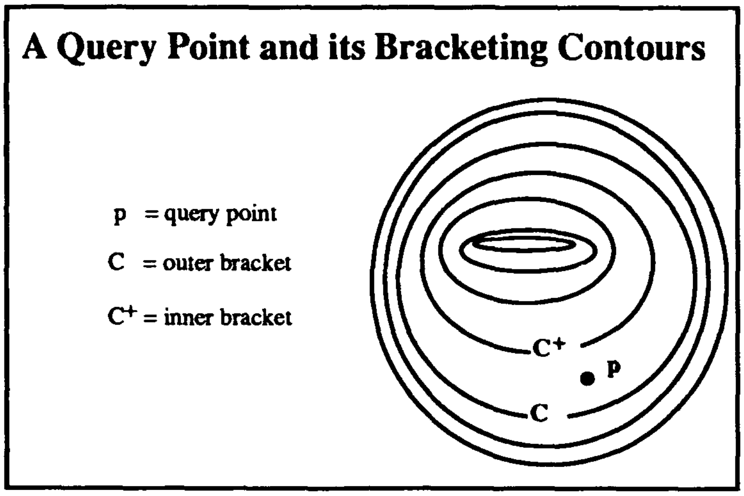

Figure 3. Bracketing a query point from within and without.

II. TWO-DIMENSIONAL BINARY SEARCH APPLIED TO TOPOGRAPHIC MAPS.

Extending binary search to two dimensions to interpret topographic maps.

Binary search has traditionally been applied to a one-dimensional data structure, sorted by some user-defined ordering property. The data structure might be an array of numbers sorted by the natural ordering of the reals, or a list of employee records sorted alphabetically by name. One commonly utilized data structure is 2D trees, in which the data consists of a set of ordered pairs of integers. In a 2D tree, the data is sorted on two keys (the abscissa and the ordinate), with one key primary. A 2D tree is not a true instance of two dimensional binary search data structure, because one key is predominant over another during the sorting process. A better candidate is outlined at [K2], in which an interior point method for linear programming "halves" an ellipse during point-in-polygon testing. However, to be truly elegant, two dimensional binary search should avail itself of the natural containment property inherent to two dimensions. In the digital domain of the computer, two dimensional objects are in general polygons. Just as the one dimensional version must check to see if a point lies between two other points, the two-dimensional version is required to decide if a polygon is contained "between" two other polygons [C1]. Betweenness is equivalent to bracketing a query point with nested polygons.

Topographical contours exhibit a natural ordering due to the way in which the forces of nature have combined to stabilize the crust of the earth. For example, gravity has assured that the top portion of a mountain has a smaller cross section than its base. Thus, when projected onto a plane, contours from the same mountain appear to be nested. Ordering by elevation, and nesting by containment are properties which may be exploited to sort contours. The data structure which results by appealing to a two-dimensional sort on elevation and nesting is called the *contour containment graph*. The motivation is that to exploit the O[log n] query power of binary search, one requires that the underlying data structure be sorted. We will see below that there are two preprocessing steps required to set up an efficient two-dimensional search of topographic maps: the first is the construction of the contour containment graph, and the second is the partitioning of the containment graph into regions suitably indexed for binary search.

The Contour Containment Graph, and Labeling of Topographic Features.

As a first step in constructing the contour containment graph, we can uniquely label each contour, and then sort all contours on elevation, in ascending order. We then "nest" contours. To illustrate, suppose a specific 100-meter contour is labeled, and the contour interval of the map is 10 meters: we now seek to find all 110 meter contours contained within the labeled contour. If we find one, we create a pointer from the 100-meter contour's label to the label of the 110 meter contour discovered to be contained in the contour. We continue this process until no more contours are found to be within the 100-meter contour. We repeat this operation for all other 100-meter contours. When this step is completed, we switch our baseline cell complex from all those bounded by 100-meter contours to all those bounded by 110-meter contours, and continue the process until there are no contours remaining to be processed. An example of a terrain and its contour containment graph is depicted in the figure below. The terrain features three hills. All three are contained within baseline contours of twenty and forty meters elevation. Note that a label may be associated with the forty meter contour to delimit the extent of the "hill country". Also, a label may be installed on each of the sixty meter contours to name the individual hills. One of the three hills contains two small knobs at the top, at an elevation of one hundred twenty meters. Each time that the set $\{C^+\}$ contains multiple elements for a given contour C, another level of sorting must be initiated to assure that the contour containment graph is properly structured and nested for binary search.

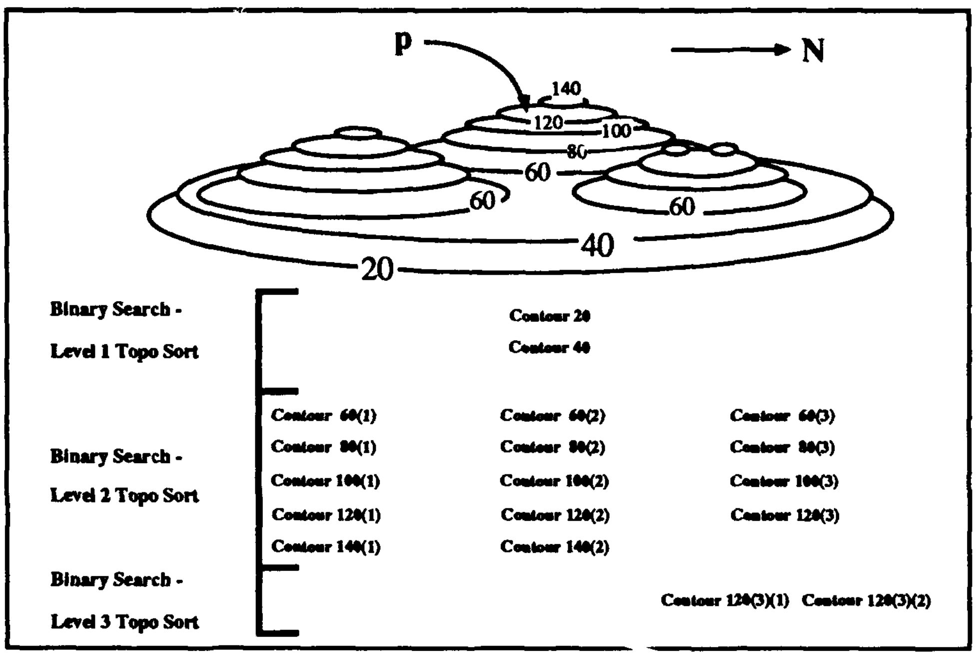

Figure 4. A sorted terrain, nested in preparation for binary search.

Within each level of a contour containment graph, binary search may be invoked to achieve O[log n] time complexity, where n is the number of contours contained at that level. To illustrate, in the figure below, a hill is represented by eight contours. On the first iteration of binary search, a contour halfway up the hillside at eighty meters is considered, and the query point is determined to be inside. On the second iteration, it is determined that the query point is not inside the one hundred twenty meter contour, which is three quarters of the way up. The third iteration decides that the point is not inside the one hundred meter contour, which is five eighths of the way to the top. At the next iteration binary search concludes, having bracketed the query point between the eighty and one hundred meter contours, while having interrogated only three of the eight contours.

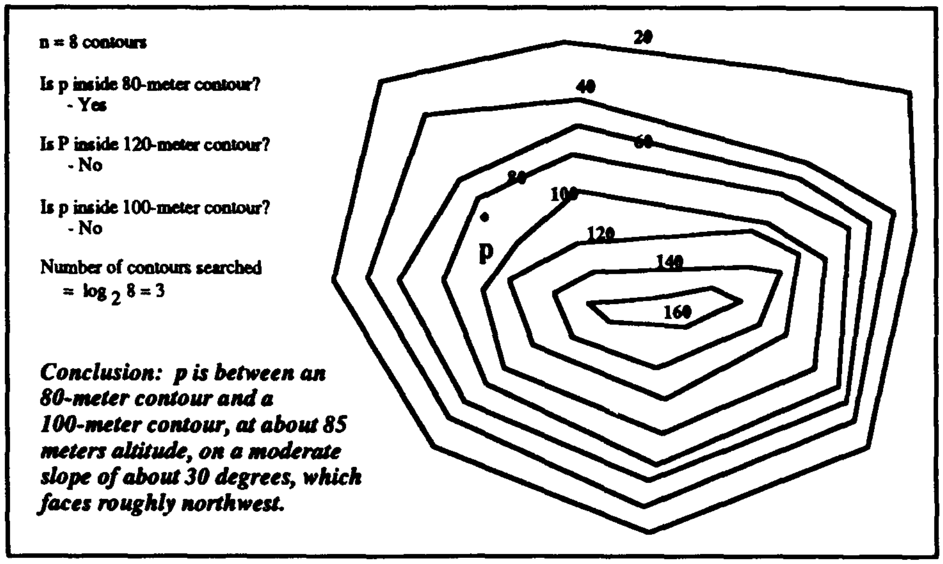

Figure 5. Binary search brackets a query point.

Although two-dimensional binary search may achieve O[log n] time complexity over a database of n contours, the issue remains open regarding the time complexity of the search as a function of the number of vertices contained within a given contour. For example, one topographic contour may contain a single vertex, whereas another may contain thousands of vertices. Processing a set of contours comprised of a small number of vertices is clearly more desirable for performance considerations than processing a set of contours comprised of a large number of vertices. An objective metric of time complexity should take into account both the number of contours and the number of vertices per contour.

Partitioning a Topographic Map for Binary Search.

Any topographical map contains contours of locally minimum elevation. These are readily identified from the contour containment graph developed in the preceding section. The strategy is to partition the map between all such contours, by constructing synthesized boundaries to act as cuts for binary search. Optimal placement of the cut boundaries is a load balancing problem, which needs to address not only the number of contours within each cut, but the total number of vertices which comprise contours in the cut. In the diagram below, four hills have been partitioned by synthesized boundaries into regions suitable for binary search. Note that the bold lines are not contours but synthetic boundaries. The first cut runs roughly down the middle of the map, and segregates the rightmost hill from the other three. Observe that the first cut contains nine contours on the left, but only seven on the right. This is not arbitrary, but is designed to compensate for the longer perimeters of the contours on the right of the cut (it is implied that a longer perimeter equates to a larger number of vertices in the contour boundary data).

The second cut is dependent upon the decision made during the first cut. If a query point is to the left of the first cut, then the second cut lies between the two most northerly hills and the hill in the southwest corner. Conversely, if the query point is to the right of the first cut, then the second cut lies halfway up the rightmost hill. Continuing in this fashion, the number three cuts are synthesized. No further cuts are shown, but the logic to create them is similar.

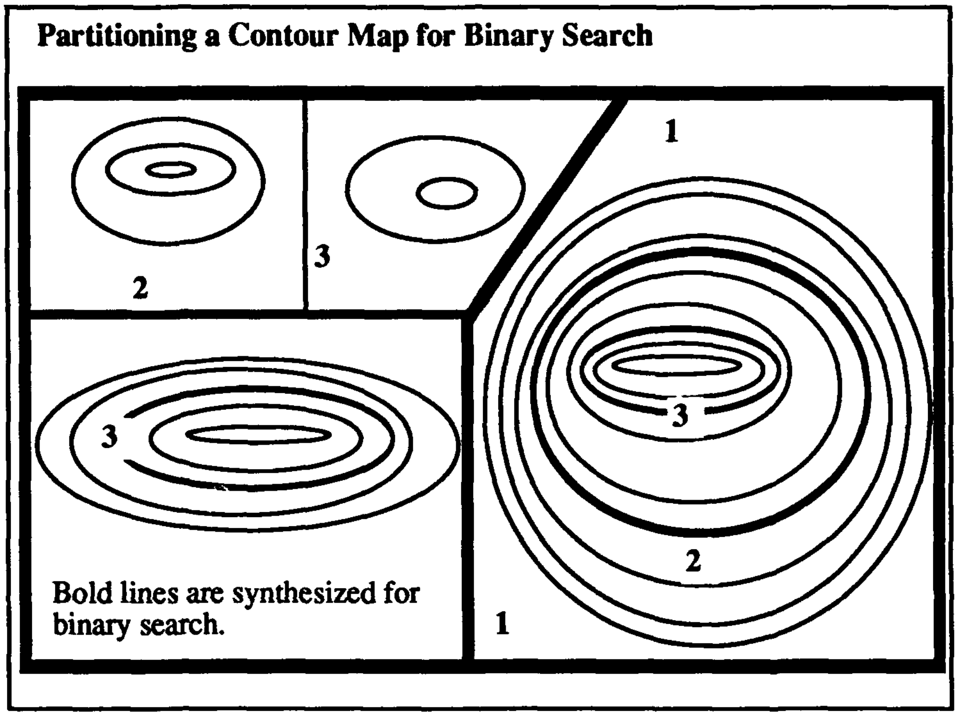

Figure 6. Load balancing a contour map to create a two-dimensional binary tree.

Dealing with contours which exit the clipping region of a map.

Figure 6 is oversimplified. In general, contours are not so well-behaved. There is one common problem to consider: a contour may exit the rectangular region bounding the map, and therefore pass outside the clipping region. The problem may be solved by conjoining the troublesome contour with the rectangular edge of the map. This contrivance forces two polygons to be synthesized from the errant contour, to create a data structure compliant with two-dimensional binary search. Synthetic boundaries for binary search may also be constructed accordingly.

The figure below depicts a clipped contour of forty meter elevation which exits the map at both sides. Two dimensional binary search requires that data structures be in the form of polygons. We synthetically create two new polygons by conjoining the clipped contour with the edges of the map. Because the point-in-polygon algorithm of choice (described in the next section) requires a sense of handedness, we assure that the vertices of the new polygons are in counterclockwise order. At execution time, we may now ask if a query point is contained within either the upper or the lower polygon manufactured by utilizing the clipped contour, and proceed accordingly.

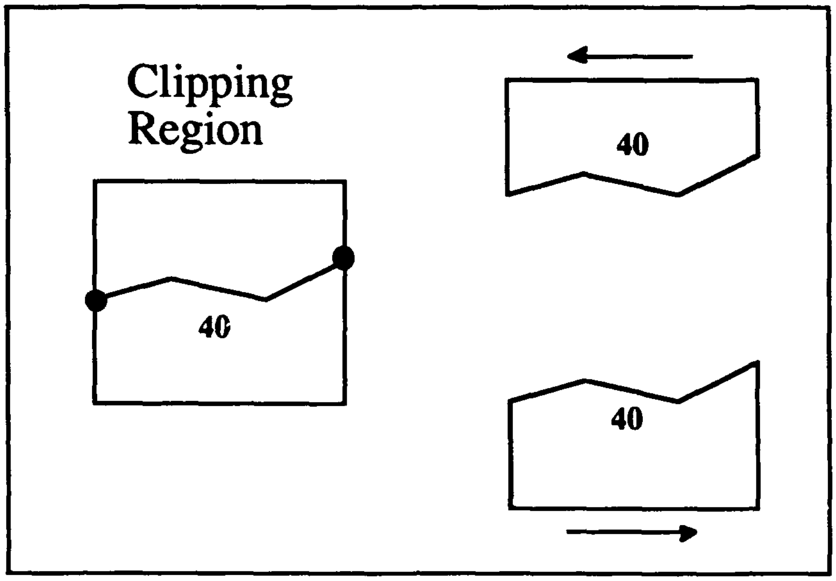

Figure 7. Creating two polygons from a clipped contour.

III. AN INCLUSION (POINT-IN-POLYGON) ALGORITHM, AND PROXIMITY.

Perceived shortcomings of currently available point-in-polygon technology.

The two-dimensional binary search algorithm requires a utility function to establish whether or not a query point is inside a topographical contour. The utility function is a true workhorse, so it must be efficient. There is no margin for error, which means that point-in-polygon algorithms which rely on the precision of machine arithmetic are inappropriate candidates. For this reason, approaches based on the winding number or the parity algorithm are currently infeasible. The Apple Macintosh family of computers has implemented a predicate called "point-in-region-p", available as part of the Quickdraw graphics repertoire, but the predicate consumes quadratic amounts of region space in memory, which becomes prohibitive for even a moderate number of polygonal boundaries. A high-performance algorithm from the computational geometry literature, based on triangulation [K3], is a viable candidate, although it remains an untested quantity, since it has never been tasked against a multi-megabyte database of topographical contours.

Because of perceived shortcomings of on-the-shelf point-in-polygon algorithms, and the lack of benchmark data to test the performance of the triangulation algorithm, the author has opted to implement his own algorithm [C2], which has been extensively tested against actual contour data. The algorithm assures that a contour is oriented in a counterclockwise direction, so that the interior of the contour is to the left during traversal. Inclusion testing is then conducted as a function of a query point's proximity to a contour (see figure below). One benefit of the algorithm is that it returns distance and direction (normal vector) to the nearest point on a boundary, in addition to the inclusion decision. As will be seen below, the normal vector is crucial to topographic map interpretation. As originally conceptualized, the algorithm anticipated that every pixel in a digital boundary would be explicitly available as part of the data structure. However, the Defense Mapping Agency does not represent feature boundaries so obviously. Instead, a contour is provided in chain-coded format, where the boundary of the contour consists of a set of ordered vertices. It is up to the user of the data to create the edges which join the vertices.

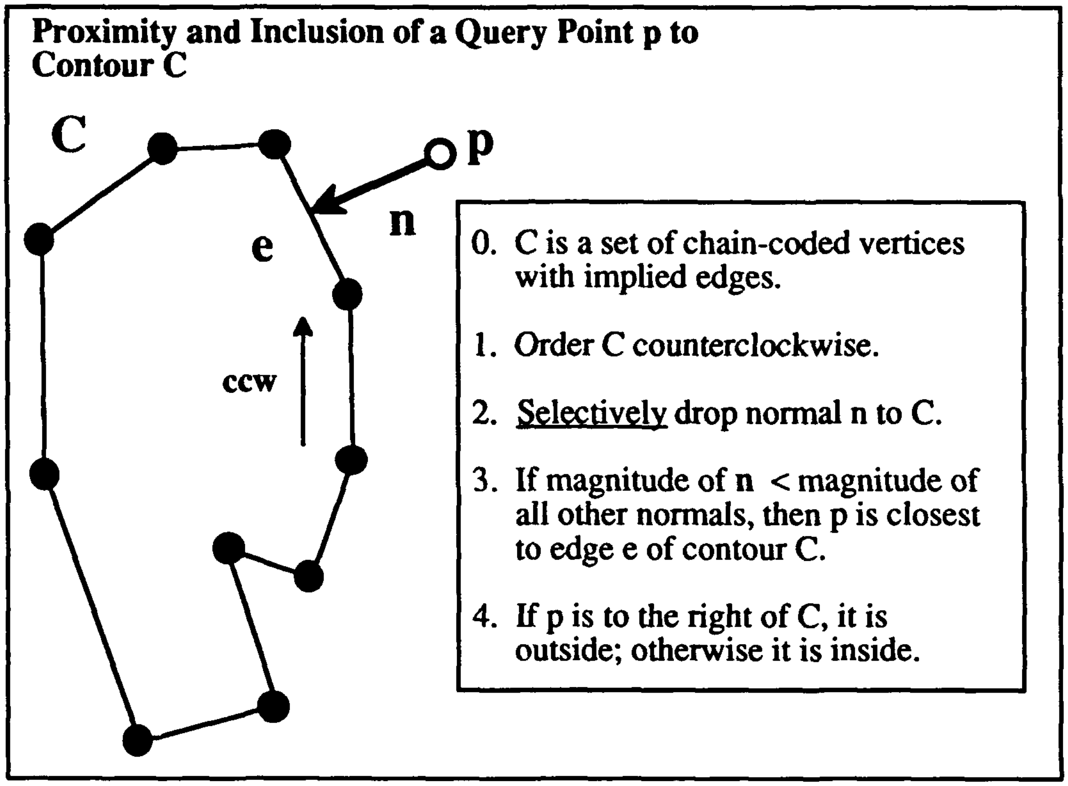

Figure 8. The normal vector may be used to decide inclusion.

The Voronoi diagram for data produced by the Defense Mapping Agency.

Vector data distributed by the Defense Mapping Agency (DMA) contains three kinds of objects: points, line segments, and polygons. It has been known for some time that the skeleton, or medial axis of a polygon consists of portions of parabolas and line segments [B2]. The parabolas are the locus of equal distance between points and segments. The line segments are the locus (angle bisectors) between extended segments. It is also true that for any set of points, segments, and polygons the equidistance locus consists of parabolas and line segments. Thus, the Voronoi diagram for DMA data, which is defined to be the locus of equal distance, is in general parabolic.

There is currently no commercial product available to generate the parabolic Voronoi diagram for an arbitrary set of polygons, segments, and points. However, there are three research and development tools (of varying degrees of robustness) circulating among researchers in academia [M3]. The developmental products implemented to date have encountered problems of numerical precision, primarily when deciding upon which side of a parabola a query point lies [F2]. However, as indicated at reference [E1], the theory behind the sweepline algorithm [F1] to generate the linear Voronoi diagram should be directly extensible to the parabolic diagram. It is clear that for the asymptotic solution to the static proximity problem, the Voronoi diagram is the paradigm of choice. As a stopgap measure, until a tool to generate the parabolic diagram is available, the author has developed his own proximity algorithm, described below, based on restricted use of the normal vector. The author's algorithm, unlike the Voronoi diagram, facilitates dynamic objects. If an object's position changes, the Voronoi diagram must reinvoke a relatively expensive preprocessing step, whereas the author's algorithm simply replaces the object's old boundary position with the new.

Finding the nearest point of a contour to a query point.

A contour, which when represented with digital data is in the form of a polygon, consists of a set of vertices and the implied edges which connect the vertices. Thus, when one speaks of proximity to a contour from a query point, one is actually referring to minimal Euclidean distance to the set of vertices, vs. distance to the set of edges.

Minimal distance to an edge is non-trivial to compute. This process entails dropping the normal vector from a query point to the edge. Since floating point operations may be required at every edge to which the normal is dropped (although the author introduces below a new technique which avoids floating point arithmetic), we would like to limit the number of edges incurring such an expensive operation. If the normal vector strikes an edge directly, the edge is said to be *admissible* to the normal vector. Refer to the figure below. Clearly, it does not behoove us to drop the normal from query point p to edge e_2, since the tip of the normal does not even intersect e_2, but rather its extension. Such cases are precisely those which we strive to avoid, by appealing to a normal vector admissibility filtering technique. It will be shown below that as a side effect, the filtering technique returns minimal distance to a vertex.

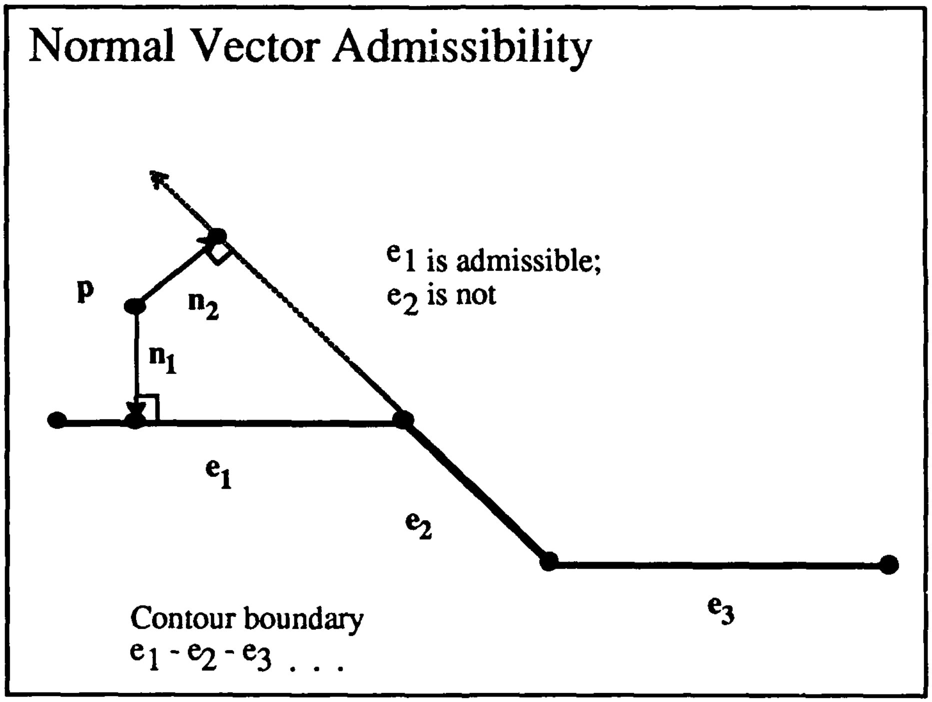

Figure 9. Certain contour edges do not admit the normal vector.

Derivation of edge admissibility conditions from the law of cosines.

Construct orthogonal rays from the endpoints of contour edge e, as in Figure 10 below. Now suppose that query point p lies between the rays. Note that the angle between edges x and e is acute, as is the angle between edges y and e. Let the angle between y and e be θ_1 and the angle between x and e be θ_2. Then, by the law of cosines,

$$x^2 = y^2 + e^2 - 2\, y\, e \cos \theta_1 \qquad [1]$$
$$y^2 = x^2 + e^2 - 2\, x\, e \cos \theta_2 \qquad [2]$$

The cosine function is positive for acute angles. We therefore obtain

$$x^2 + \alpha_1 = y^2 + e^2 \qquad [3]$$
$$y^2 + \alpha_2 = x^2 + e^2; \qquad \alpha_1, \alpha_2 \geq 0 \qquad [4]$$

These equations are alternatively expressed by the inequalities:

$$x^2 \leq y^2 + e^2 \qquad [5]$$
$$y^2 \leq x^2 + e^2 \qquad [6]$$

This set of inequalities must be true for segment e to admit the normal vector. Point p of Figure 10 satisfies the conditions.

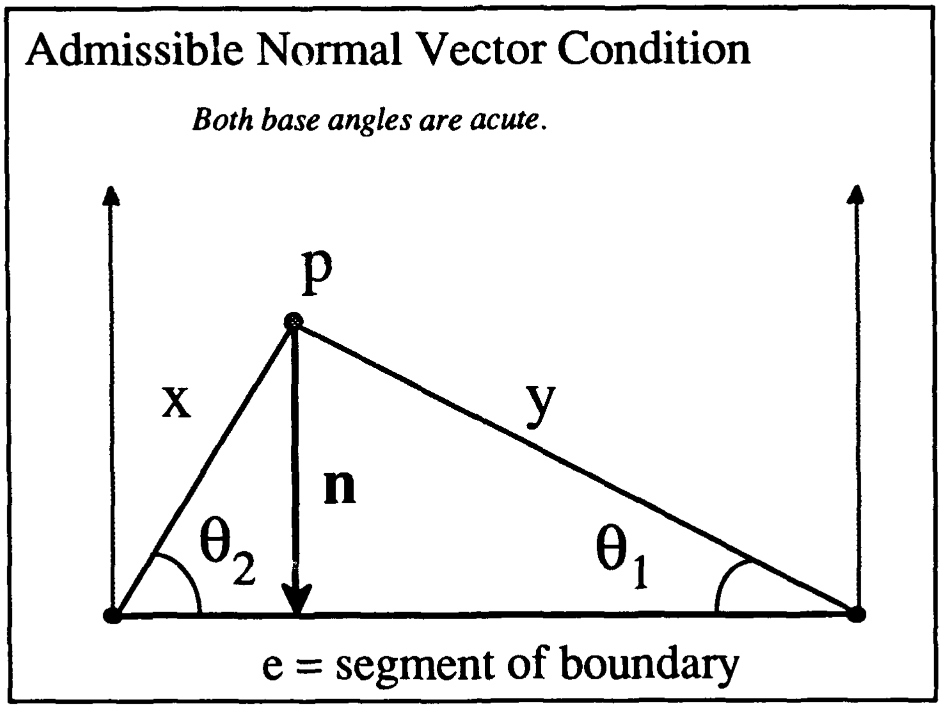

Figure 10. An edge is admissible if base angles are both acute.

In practice, it is more likely for the test to be failed than to be passed, so it makes sense to test first for failure rather than for success. The failure condition may be written as the predicate

$$\neg\, [\; x^2 \leq y^2 + e^2 \quad \wedge \quad y^2 \leq x^2 + e^2 \;] \qquad [7]$$

From DeMorgan's rules, this may be rewritten

$$\neg [x^2 \leq y^2 + e^2] \quad \vee \quad \neg [y^2 \leq x^2 + e^2] \qquad [8]$$

which is equivalent to

$$x^2 > y^2 + e^2 \quad \vee \quad y^2 > x^2 + e^2 \quad . \qquad [9]$$

If either side of disjunction [9] is true, then edge e is not admissible to the normal vector, and a potentially expensive floating point operation is avoided by means of a simple integer-valued decision function. An example of satisfaction of the second inequality of the disjunction is illustrated at the figure below. In this case, edge e fails the admissibility condition, so that the normal vector computation is avoided.

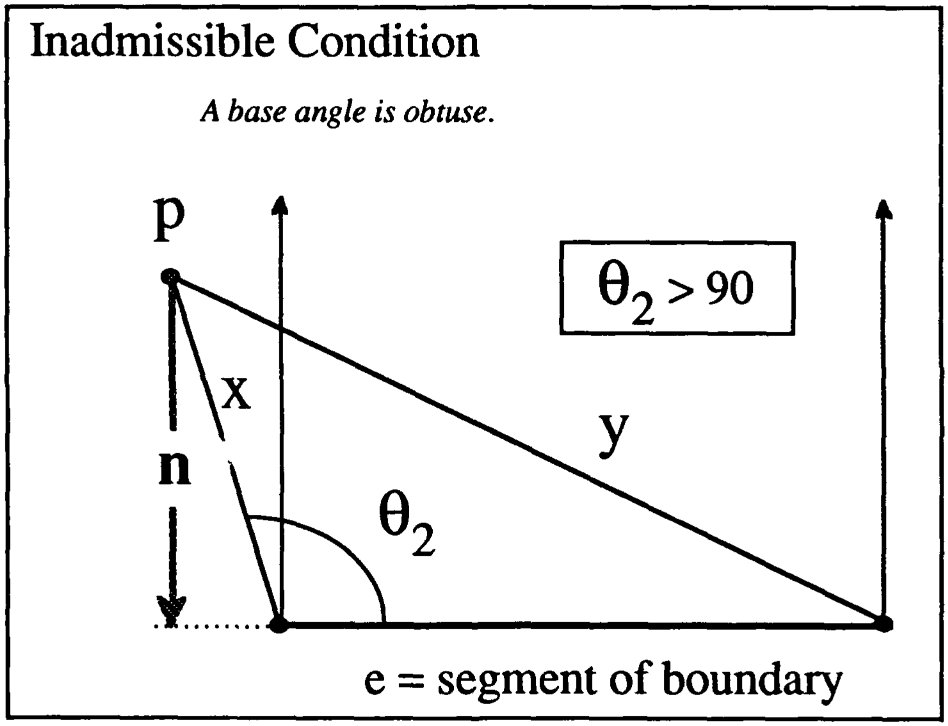

Figure 11. An obtuse base angle precludes admissibility.

As a byproduct of the admissibility test, minimal distance to a vertex is returned. Consider the integer-valued expression $(s_p-s_v)^2 + (t_p-t_v)^2$, where (s_v,t_v) is the coordinate at the vertex and (s_p,t_p) is the coordinate at the query point. This expression is synonymous with either of the arguments x^2 or y^2 in equations [1]-[9] above. Hence, the filtering operation as a side effect monitors the squares of the distances to each of the vertices of a contour. When the smallest such expression is found across all vertex possibilities, the square root is extracted. The entire process involves n integer-valued operations for n vertices, and one floating point operation, for a time complexity of O[n]. The integer-valued operation here involves two integer multiplies, three integer adds, and an integer comparison. The floating point operation is a single-shot appeal to the square root of the minimal integer-valued result.

A Common Lisp implementation of the edge admissibility test might appear as follows:

```
(defun admissible-normal-segment-p (x y ax ay bx by)
; (ax, ay) and (bx, by) are the endpoints of segment e in figures; (x,y) is query point.
        (prog (dis1sqr dis2sqr dis3sqr)
        (declare (type longint x y ax ay bx by dis1sqr dis2sqr dis3sqr))
        dis1sqr = (dissqr ax ay bx by)
        dis2sqr = (dissqr x y ax ay)
        dis3sqr = (dissqr x y bx by)
        (cond  ((> dis3sqr (+ dis1sqr dis2sqr))(return nil))
               ((> dis2sqr (+ dis1sqr dis3sqr))(return nil))
               (t (return t)))))
```

Finding the normal vector with minimum magnitude, across all segments.

Although we now have a test to determine which segments of a boundary admit the normal vector from a query point, we have not said anything about the actual computation of the minimal such vector across all segments. In this section we develop a new test to find the smallest normal vector, without resorting to any floating point computations. If the actual magnitude is desired, two floating point operations are required over the entire database. We appeal to a very useful result from analytic geometry, called the Cevian formula (for a development see [K1]). A *cevian* is defined to be a line segment drawn from a vertex of a triangle to the opposite side. Note that medians, angle bisectors, and altitudes are all examples of cevians. The Cevian formula is shown in the figure below, where n is an altitude in this case. It is convenient that the altitude is equivalent to the normal vector under discussion. In the figure, observe that rz and sz are lengths which sum to side z, whereas r and s are ratios which sum to one.

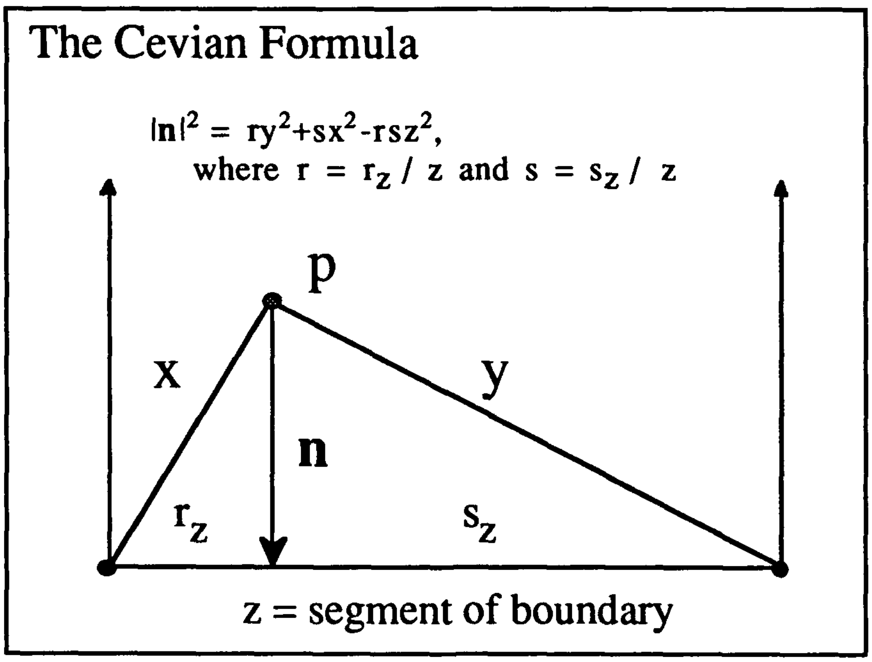

Figure 12. The Cevian formula relates a normal vector to the sides of a triangle.

Unfortunately, we do not know the values of r and s, because we do not know the point at which the normal vector impacts side z. In the equations below, which until step [15] echo the discussion in [K1], we derive a formula for the square of the magnitude of the normal in terms of a ratio involving the squares of the sides. Steps [10]-[11] are a reiteration of the information conveyed by the figure. Steps [12]-[13] involve a substitution for s, followed by a reformulation as a quadratic equation in terms of r. In step [14] we set the discriminant equal to zero, because the roots of equation [13] must be non-negative and equal, since r and s form a convex set. Solving this equation for n^2 results in the ratio shown at [15], which but for the divide operation is economical to compute, since it involves four integer multiplies and three adds. If one were tuning the technique with assembly code, two of the multiplies (those involving the 4) could be converted into two-bit left shifts, since shifts are cheaper than multiplies.

$$r = \frac{r_z}{z};\ s = \frac{s_z}{z};\ r + s = 1;\ r_z + s_z = z \qquad [10]$$

$$n^2 = ry^2 + sx^2 - rsz^2 \qquad [11]$$

$$n^2 = ry^2 + (1-r)x^2 - r(1-r)z^2 \qquad [12]$$

$$z^2r^2 - (x^2 + z^2 - y^2)r + (x^2 - n^2) = 0 \qquad [13]$$

$$(z^2 - y^2 + x^2)^2 - 4z^2(x^2 - n^2) = 0 \qquad [14]$$

$$n^2 = \frac{4x^2z^2 - (x^2 + z^2 - y^2)^2}{4z^2} \qquad [15]$$

What about the division by $4z^2$, which implies a floating point operation? The answer is that in order to find the normal vector of smallest magnitude, we may refrain from performing the division until all admissible segments have been associated with a numerator and denominator as at [15], and checked against the shortest normal vector found thus far. The check is made as follows. Let n_1 be the normal dropped from a query point to segment z_1, with x_1 and y_1 the distances to the respective endpoints of z_1. Let n_2, z_2, x_2, and y_2 be defined similarly. Then the squares of the normals are shown respectively at [16] and [17]. Now $n_1 < n_2$ if and only if [19] and [20] are true, but [20] is true if and only if the product of the means is less than the product of the extremes as shown at [21]. Cancelling the common factor produces test inequality [22]. Notice that if we are using integer-valued coordinates, as we must if we are working with data displayed to a computer screen, there are no floating point expressions involved in the test.

$$n_1^2 = \frac{4x_1^2z_1^2 - (x_1^2 + z_1^2 - y_1^2)^2}{4z_1^2} \qquad [16]$$

$$n_2^2 = \frac{4x_2^2z_2^2 - (x_2^2 + z_2^2 - y_2^2)^2}{4z_2^2} \qquad [17]$$

$$n_1 < n_2 \Leftrightarrow \qquad [18]$$

$$n_1^2 < n_2^2 \Leftrightarrow \qquad [19]$$

$$\frac{4x_1^2z_1^2 - (x_1^2 + z_1^2 - y_1^2)^2}{4z_1^2} < \frac{4x_2^2z_2^2 - (x_2^2 + z_2^2 - y_2^2)^2}{4z_2^2} \Leftrightarrow \qquad [20]$$

$$\left[4x_1^2z_1^2 - (x_1^2 + z_1^2 - y_1^2)^2\right]4z_2^2 < \left[4x_2^2z_2^2 - (x_2^2 + z_2^2 - y_2^2)^2\right]4z_1^2 \Leftrightarrow \qquad [21]$$

$$z_2^2\left[4x_1^2z_1^2 - (x_1^2 + z_1^2 - y_1^2)^2\right] < z_1^2\left[4x_2^2z_2^2 - (x_2^2 + z_2^2 - y_2^2)^2\right] \qquad [22]$$

Using the test is simple. As input we receive a query point and candidate segment with integer coordinates. The squares of the distances from the query point to the segment endpoints are computed with the usual Euclidean formula, as is the square of the distance between the endpoints. These three quantities are used to compute the integer-valued numerator and denominator of equation [15]. The same technique applied to some other candidate segment produces another numerator and denominator, which we cross-multiply with the first at inequality [22]. If the product of the means is less than that of the extremes, then the first segment is closer to the query point; otherwise the second segment is closer. We continue this process until all segments are exhausted, remembering the segment giving rise to the shortest normal vector as we do so.

Observe that we have located the nearest segment (according to the true Euclidean metric) to a query point without resorting to any floating point arithmetic. Granted, we do not yet know the magnitude of the shortest normal vector, but we know that we have the shortest. To obtain the magnitude, we merely need to perform the division indicated at equation [15], and extract the square root of the result. Note also that we never had to compute any of the points of a line segment; we were able to make do with the vertices at its endpoints. This latter artifact demonstrates the power and leverage of the Cevian formula, developed over three centuries ago. The formula may potentially be used to assist in the generation of the parabolic Voronoi diagram for line segments and polygons.

We briefly summarize before moving on to the next section. When the two-dimensional binary search paradigm requests the inclusion algorithm to decide whether or not a query point is contained within a specific contour, the inclusion algorithm is handed the counterclockwise-oriented set of contour vertices and the query point as arguments. The first action taken by the inclusion algorithm is to subject all of the implied edges of the contour to the normal vector admissibility test, maintaining the squared distances to the vertices on the side. Generally, the test returns just a handful of edges admissible to the normal vector. To each of these, the cross-product test shown at [22] is performed to locate the minimal normal vector. This quantity is compared against the minimal result obtained for the vertices. If the square of the distance to an edge is smaller than the squared distance to a vertex, a test is invoked to decide if the query point is to the left or the right of the edge; if to the left, the point is inside the contour, and if to the right, the point is outside. At this time the numerator and denominator of equation [15] may be divided and the square root extracted to obtain the actual magnitude of the normal vector. If the squared distance to a vertex is smaller than that to an edge, a synthetic edge is constructed from the vertex's predecessor and successor vertices in the contour boundary, and a test is invoked to decide if the *vertex* is to the left or the right of the synthetic edge; if to the left, the query point is inside the contour, and if to the right, the point is outside. The square root may be extracted to obtain the magnitude of the normal vector. The shortest normal vector points to either the inner or the outer bracketing contour of the query point.

IV. INTERPRETATION OF A QUERY POINT IN THE CONTEXT OF A MAP.

Binary search of a contour containment graph concludes by returning the two bracketing contours of a query point. The algorithm is now armed with all the information it requires to produce an "interpretation" of a query point, as defined in the first section of the paper. If either bracket has inherited the name of a mountain, hillside, crater, etc., for which the bracket is a structural element, then the name is available for simple display, or for further spatial reasoning operations such as line-of-sight or traversibility reasoning. Because inclusion testing as described above returns as a byproduct the normal vector from a query point to a contour, both the distance to the outer bracket and the distance to the inner bracket are known when binary search completes. These two distances may be used in conjunction with the contour interval of the map to obtain estimates for the point's elevation and slope. The direction from a hilltop to the query point, together with the elevation values and orientation of the bracketing contours, may be used to determine a directional gradient, which establishes upon which flank of a hillside a query point resides. The details involved in extracting the elevation, the slope, and the directional gradient are described below.

Deriving the elevation of a query point from its bracketing contours.

Once the bracketing contours for a query point have been established, it is a simple matter to compute an interpolated elevation at the query point. Without loss of generality, let us assume p is on an uphill slope from outer bracket C to inner bracket C^+, as depicted at the figure below. The elevation of query point p, denoted El(p), may be obtained by using similar triangles to compute an expression which accounts for p's relative location between the contours, and multiplying it by the fixed contour interval of the map. To this expression is added the baseline elevation at p's outer bracket (if p were on a downhill slope, the expression would be subtracted instead). Special cases require additional processing. If a query point has an outer bracket but no inner bracket, as it will when it resides within a contour which represents a hilltop, and there are no control points available to indicate the actual elevation at the hilltop, then the query point inherits the elevation of its outer bracket, since interpolation is impossible. If a control point is available (generally obtained by surveyors with a spirit level, and represented on the map with an "X" or a delta symbol), then interpolation is possible even in the absence of an inner bracketing contour. One simply coerces the inner bracketing contour to be the control point, and temporarily sets the map contour interval to be the difference between the elevation of the control point and the elevation of the outer bracket. Downhill slopes, craters, saddles, and culverts may be treated with similar logic.

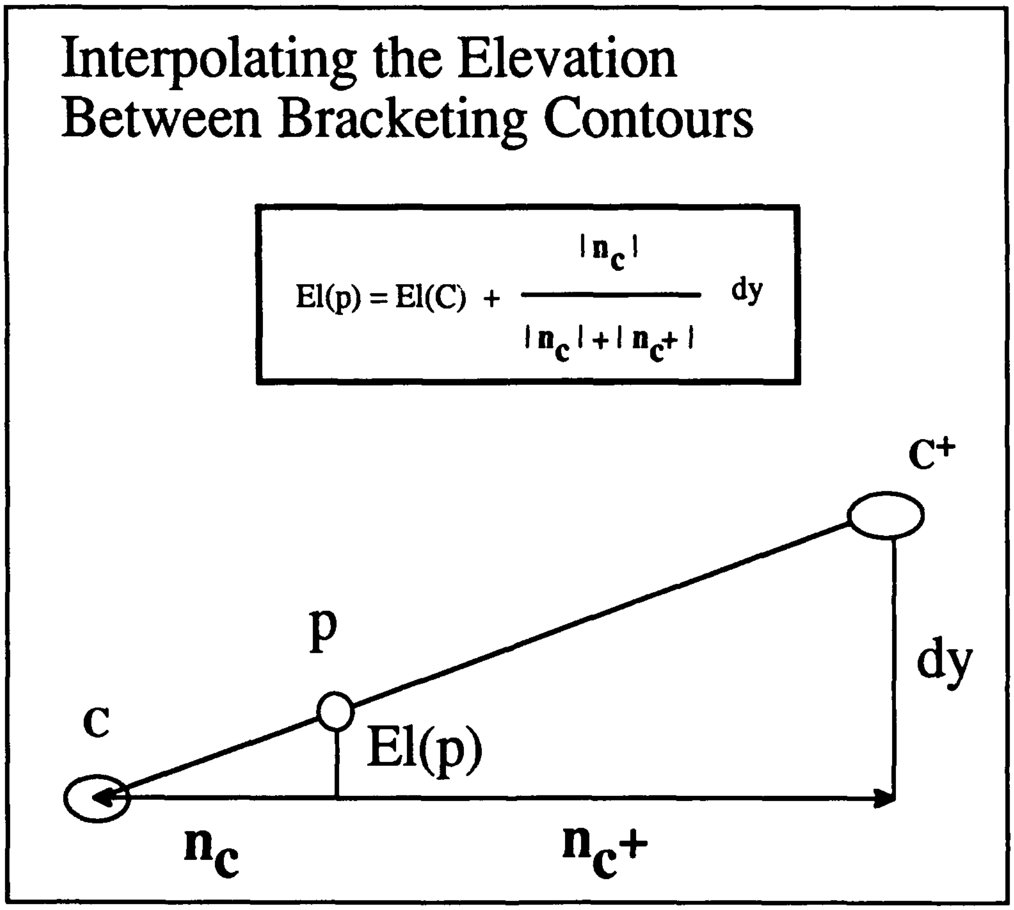

Figure 13. Elevation is obtained through simple linear interpolation.

Obtaining the slope at a query point from its bracketing contours.

The local slope at a query point is so simple to estimate that even interpolation is not required. It is simply the angle with a tangent equal to "the rise over the run". The "rise" is fixed, as it is given by the contour interval of the map. The "run" is defined to be the sum of the magnitudes of the normal vectors drawn to the outer and inner bracketing contours. At the top of a hill or at the base of a depression, in a saddle or a culvert, the slope is assumed to be zero, for flat ground. However, if a control point is available to provide additional elevation data, then logic similar to that outlined for elevation in the paragraph above may be utilized to obtain a refined estimate of slope. Outside the limits of the lowest lying contours, the algorithm is designed to return the string "drainage area", which again is assumed to be flat ground. There may or may not be a perennial stream flowing through a drainage area, but during flashfloods it is assumed that water would flow there.

Note that a peculiar thing happens if we slide the query point along either of the normal vectors pointing to the bracketing contours. The slope remains fixed as we do so. This is the price we pay for approximating a terrain by a set of cross-sectional contours. The computed slope cannot be made more accurate than the resolution imposed by the contour interval of the map. Thus, between any two nested contours, there is a vector field of slope vectors which connect every digital point of the inner bracket with some digital point of the outer bracket, and vice versa.

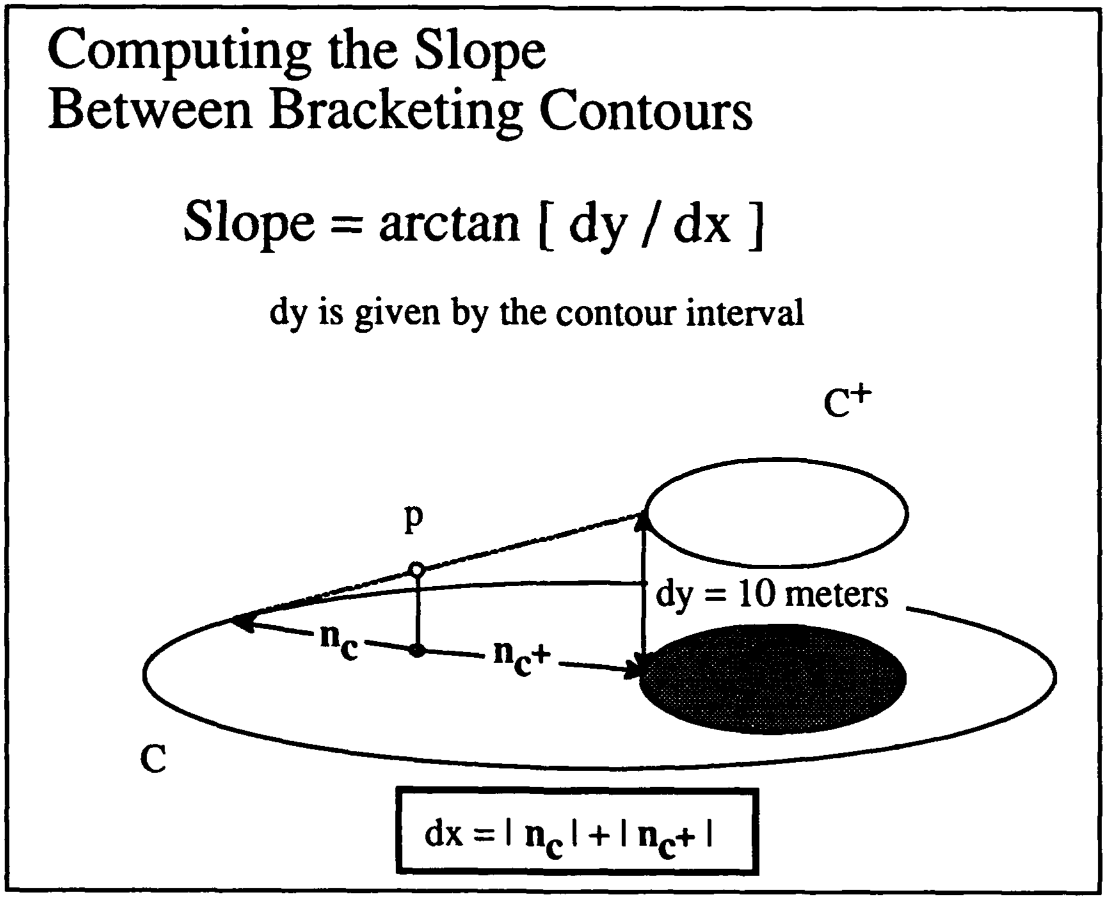

Figure 14. The "rise" is fixed, and the "run" is the sum of the normal vector magnitudes.

Obtaining the directional gradient at a query point from its bracketing contours.

Again, assume the familiar example of an uphill slope, so that El(C) = El(C^+) - k, where k is the fixed contour interval of the map. Construct the vector from the hilltop to the query point. Define the hilltop to be the control point at the top of the hill if it exists; otherwise make it some reasonable estimate, such as the centroid of the topmost contour. If there are multiple topmost contours, then make the hilltop the centroid of them all. Suppose that the hilltop to query point vector points to the left, as in the figure below. Then it is pointing downhill, because C's elevation is less that that of C^+, and it is pointing to the west, since due north is as shown by the map. The query point is therefore on the western flank of a hillside. Variations on this theme are computable for other configurations of terrain. If the elevation of C is greater than that of C^+, and we observe a leftward-pointing vector, then we would be on the western flank of a crater or valley. If the elevation of C was to be equal to that of C^+ and the vector was to point to the south, then the query point would be on a saddle or in a culvert, oriented in an east-west fashion.

The vector pointing from a hilltop to a query point is a suitable gauge of directional gradient from a global perspective. However, a query point may be situated locally on a geologic feature of a hillside, with an orientation seemingly at odds with the global result. For example, on the south side of a mountain, there may be a ridge which proceeds from the summit down to the south. The ridge will have both eastern and western flanks. Suppose for the sake of argument that a query point is on the western flank of the ridge. We conclude it is possible for a query point to be locally on a western flank, but globally on the south side of the mountain. The local flank estimate is easily computed by drawing the normal vector from the query point to its outer bracketing contour. The vector points in the compass direction of the local gradient. This procedure is particularly useful for rugged terrain such as that encountered on Mount Rainier in the state of Washington, where contour data tends to resemble a set of nested "octopuses".

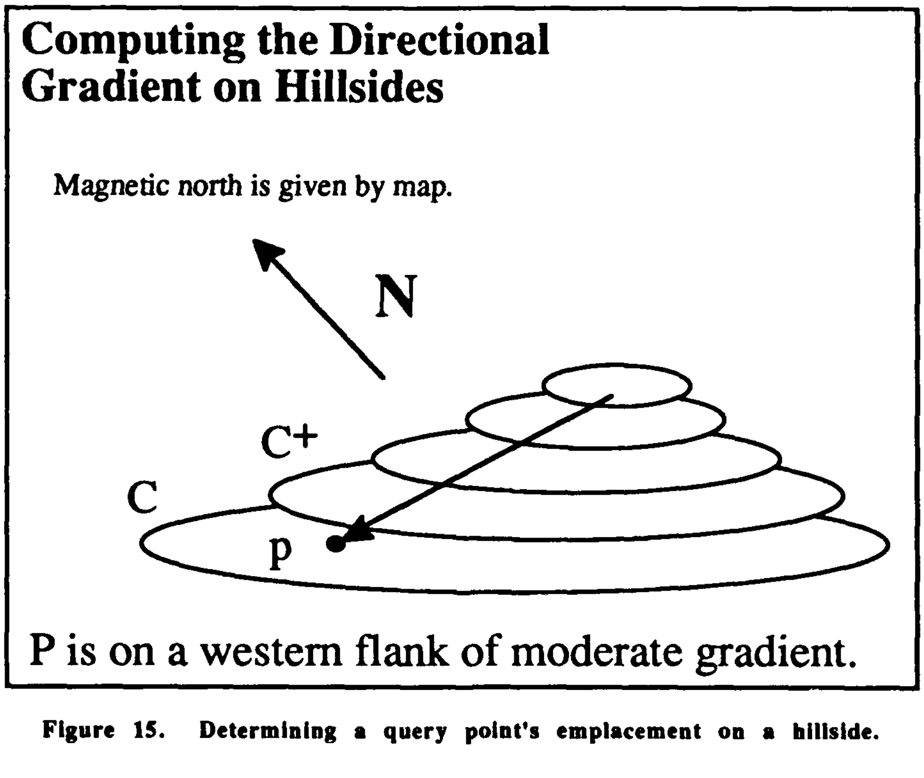

Figure 15. Determining a query point's emplacement on a hillside.

V. AN IMPLEMENTATION, AND CONCLUSIONS.

The theory of automated topographic map interpretation, as developed to date, has been partially implemented on a Macintosh IIfx workstation, using Macintosh Common Lisp, version 2.0. There are plans to convert the code into C, using the Symantec Think C environment. The conversion is intended not so much for performance purposes, since the Lisp compiler's performance is favorable when compared to that of the Symantec package, but to be able to control the process of garbage collection, which in the Lisp package is beyond the reach of the user.

There are two databanks of contour data: real and simulated. The first source of the real variety is a set of digital elevation matrix (DEM) data, which is a gridded representation of elevation values sampled at equispaced x and y increments. DEM data is produced by the United States Geographical Survey (USGS) office, as a result of data collection performed primarily by civilian engineers. To obtain topographical contours from DEM data, one may utilize a geographic information system (GIS) to extract contiguous points of equal elevation from the grid. The author used an on-the-shelf GIS package called Macgrizo [M1] to create contours for the Killeen Texas area. The second source of real data, which is the military counterpart to DEM data, is digital terrain clevation data (DTED), which currently is available in two resolutions: Level I, at 100 meter spacing, and Level II, at 30 meter spacing.

The simulated data is handcrafted by appealing to Macintosh Quickdraw graphics. A representative terrain containing four hillsides is depicted in the figure on the next page, where a query point is represented by the tip of the cursor (the arrowhead at the right). In this case, the partitioning algorithm during a preprocessing step created level one and level two cuts to segregate the four hillsides. The level three cuts and beyond partitioned each of the individual hills, using the nesting principle described earlier. Now comes execution time, and two-dimensional binary search. In this example, hills two and four were selected in the first binary cut, and hill two in the second cut. In the third cut, it was determined that the query point was not inside hill two's forty meter contour; in the fourth cut it was determined that the point was inside the twenty meter contour. Binary search concludes at this time because the contour containment graph has been exhausted. Therefore, the outer bracket is hill two's twenty meter contour, and the inner bracket is the forty meter contour. Associated with each of these two contours is the label "HILL2". The gradient computation deduces an easterly downhill slope; the slope is computed to be forty eight degrees; and the elevation interpolates to thirty three meters. Currently, the interpretation process says nothing about the relationship among HILL2 and the other hills; future work will address this issue.

Future Work.

The research to date has focused on a local interpretation of a query point. By definition, a local interpretation is limited to a description of a query point in terms of the label of the hill upon which it resides, the two contours which bracket it, an interpolated elevation, a slope value, and a directional gradient. This information is useful for localized reasoning about the immediate environs of a query point. A natural outgrowth of this work is to extend the reasoning to a more global capability. For example, one could utilize knowledge about the location of a hillside with respect to other hillsides in a specific region, to achieve context-cued line of sight reasoning or traversibility planning.

To illustrate line-of-sight reasoning, consider the following example, based on the author's personal experience. In Grand Teton National Park in Wyoming, if one is on the western shore of Jenny Lake, the tallest visible peak is Teewinot Mountain, which looms spectacularly nearly a mile above the observer's head. One peak away is the Grand Teton, which although a thousand feet higher, may not even be seen from this vantage point because it is blocked from sight by Teewinot. The interpretation process described in Section IV would determine that the query point is on the eastern flank of Teewinot Mountain, on a moderate slope, at about 6600 feet elevation. Utilization of the directional gradient calculation would indicate that the direction to the tops of Teewinot and the Grand Teton are roughly the same, but the slope of the segment joining the query point to the top of Teewinot is greater than that drawn to the top of the Grand Teton. Hence, one concludes that line-of-sight westward to the Grand is restricted by the intervening mass of Teewinot. Future work will involve refining and formalizing concepts such as these.

Already, the normal vector admissibility filtering technique has been extended to objects other than topographic contours. The Defense Mapping Agency produces a set of vector overlays corresponding to a transportation network, a hydrology network, obstacles, surface orientation, surface composition, and vegetation type. In additicn, the DMA produces a gridded product called digital terrain elevation data (DTED), at both thirty and one hundred meter horizontal resolution. The vector products together with thirty-meter DTED in large part comprise what is known as tactical terrain data (TTD), a database being developed by DMA with the cooperation of the US Army Topographic Engineering Center [M2]. The integer-based decision rule derived from the law of cosines has proven to be of high utility in gauging proximity and inclusion with respect to the multi-megabyte vector databases contained in TTD.

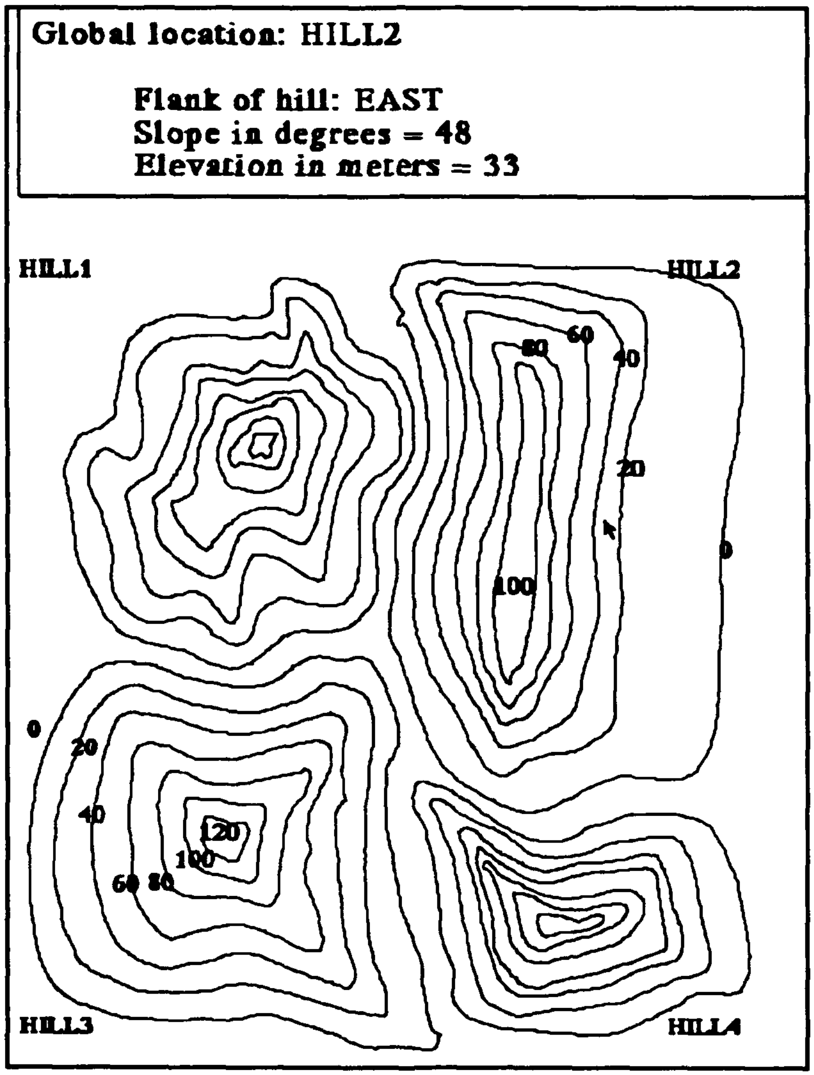

Figure 16. Interpreting a query point in terrain.

Conclusions.

Two-dimensional binary search has been utilized in conjunction with two new algorithms which avoid the expensive floating point operations associated with computing the normal vector, to produce an algorithm adept at locally interpreting topographic line maps. An interpretation consists of a human-like description of a query point in terms of its global location, interpolated elevation, local slope, and directional gradient. The search algorithm relies heavily upon proximity and inclusion algorithms developed with computational geometry research funded by the US Army. For credibility, the technique is being leveraged against multi-megabyte databases of contour information corresponding to actual terrain. An integer-based decision function which arbitrates when to drop the normal vector to an edge (during proximity testing) has proven to be extensible to objects other than elevation contours, such as segments of roads and streams, and polygons delimiting types of vegetation cover and surface material composition. As a byproduct of the research, an algorithm based on the ancient Cevian formula has been developed to find the nearest segment to a query point, without using any floating point operations whatsoever. New work will focus on extending the definition of map interpretation to be more globally descriptive of a terrain.

Bibliography

[B1] Bentley, J., Programming Pearls, Addison-Wesley Publishing Company, Reading MA, 1986.

[B2] Blum, H., A Transformation for Extracting New Descriptors of Shape, in Symp. Models for Perception of Speech and Visual Form, MIT Press, 1967.

[C1] Cronin, T., Topographical Contour Betweenness Testing, US Army Signals Warfare Center Technical Report CSW-88-7, 1988.

[C2] Cronin, T., Optimized Annulus-based Point-in-Polygon Inclusion Testing for d Dimensions, Transactions of the Seventh Army Conference on Applied Mathematics and Computing, West Point NY, 1989.

[E1] Edelsbrunner, H., Algorithms in Combinatorial Geometry, Springer-Verlag, Berlin Germany, 1987.

[F1] Fortune, S., A Sweepline Algorithm for Voronoi Diagrams, Proceedings of the Second Annual ACM Computational Geometry Symposium, 1986.

[F2] Fortune, S., private communication, AT&T Bell Laboratories, March 1992.

[K1] Kay, D., College Geometry, Holt, Rinehart, and Winston, Inc., New York NY, 1969.

[K2] Khachiyan, L.G., A Polynomial Algorithm in Linear Programming, **Soviet Math. Dokl., Vol 20, No.1**, 1979.

[K3] Kirkpatrick, D.G., Optimal Search in Planar Subdivisions, **SIAM J. Comp. 12**, 1983.

[K4] Kjellstrom, B., Be Expert with Map and Compass, rev. ed., Charles Scribner's Sons, New York NY, 1967.

[M1] Macgridzo: The Contour Mapping Program for the Macintosh, Users Manual for Version 3, Rockware Inc., Wheat Ridge CO, 1990.

[M2] Messmore, J. and L. Fatale, Phase I Tactical Terrain Data (TTD) Prototype Evaluation, US Army Engineer Topographic Laboratories Technical Report ETL-SR-5C, Ft. Belvoir VA, December 1989.

[M3] Mitchell, J., private communication, SUNY Stony Brook, July 1992.

[T1] Tech-Tran, Vol. 13, Num. 4, US Army Engineer Topographic Laboratory, Ft. Belvoir VA, Fall 1988.